Northrop Grumman
Raytheon

DOC • DoD
NASA

NATIONAL POLAR-ORBITING OPERATIONAL ENVIRONMENTAL SATELLITE SYSTEM (NPOESS)

OPERATIONAL ALGORITHM DESCRIPTION DOCUMENT FOR COMMON GEOLOCATION (D41869 Rev A)

CDRL No. A032

Northrop Grumman Space & Mission Systems Corporation
One Space Park
Redondo Beach, California 90278

Copyright © 2004-2010
Northrop Grumman Corporation and Raytheon Company
Unpublished Work
ALL RIGHTS RESERVED

Portions of this work are the copyrighted work of Northrop Grumman and Raytheon. However, other entities may own copyrights in this work.

This documentation/technical data was developed pursuant to Contract Number F04701-02-C-0502 with the US Government. The US Government's rights in and to this copyrighted data are as specified in DFAR 252.227-7013, which was made part of the above contract.

This document has been identified per the NPOESS Common Data Format Control Book – External Volume 5 Metadata, D34862-05, Appendix B as a document to be provided to the NOAA Comprehensive Large Array-data Stewardship System (CLASS) via the delivery of NPOESS Document Release Packages to CLASS.

The information provided herein does not contain technical data as defined in the International Traffic in Arms Regulations (ITAR) 22 CFR 120.10.

This document has been approved by the Unites States Government for public release in accordance with NOAA NPOESS Integrated Program Office.

Distribution: Statement A: Approved for public release; distribution is unlimited.

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

NATIONAL POLAR-ORBITING OPERATIONAL ENVIRONMENTAL SATELLITE SYSTEM (NPOESS)

OPERATIONAL ALGORITHM DESCRIPTION DOCUMENT FOR COMMON GEOLOCATION (D41869 Rev A)

PREPARED BY:

Ronson Chu AM&S Algorithm Lead	Date
Paul D. Siebels IDPS PRO SW Manager	Date

ELECTRONIC APPROVAL SIGNATURES:

Roy Tsugawa Algorithm & Data Processing IPT Lead & Algorithm Change Control Board Chairperson	Date
Gerald J. Mulvey Senior Systems Engineer	Date

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

Revision/Change Record

Document Number **D41869**

Revision	Document Date	Revision/Change Description	Pages Affected
---	8-01-05	Reflects Science To Operational Code Conversion - Initial Release.	All
A	4-22-09	Incorporated TIM comments and prepared for ARB/ACCB submittal. Incorporates ECR A-234. Approved for Public Release per Contracts Letter 091201-02.	All

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)

PDMO

Table of Contents

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

List of Figures

List of Tables

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

1.0 INTRODUCTION

1.1 Objective

The purpose of the Operational Algorithm Description (OAD) document is to express, in computer-science terms, the remote sensing algorithms that produce the National Polar-Orbiting Operational Environmental Satellite System (NPOESS) end-user data products. These products are individually known as Raw Data Records (RDRs), Temperature Data Records (TDRs), Sensor Data Records (SDRs) and Environmental Data Records (EDRs). In addition, any Intermediate Products (IPs) produced in the process are also described in the OAD.

The science basis of an algorithm is described in a corresponding Algorithm Theoretical Basis Document (ATBD). The OAD provides a software description of that science as implemented in the operational ground system --- the Data Processing Element (DPE).

The purpose of an OAD is two-fold:

1. Provide initial implementation design guidance to the operational software developer
2. Capture the “as-built” operational implementation of the algorithm reflecting any changes needed to meet operational performance/design requirements

An individual OAD document describes one or more algorithms used in the production of one or more data products. There is a general, but not strict, one-to-one correspondence between OAD and ATBD documents.

1.2 Scope

The scope of this document is limited to the description of the core operational algorithm(s) used by all the sensors to create the specific sensor’s geolocation Products. Because the CMN GEO library of functions was derived from the VIIRS SDR Geolocation module (science code), no CMN GEO Algorithm Theoretical Basis Document (ATBD) was provided. Therefore the theoretical basis for this algorithm is described in Section 3.3 of the VIIRS Geolocation Algorithm Theoretical Basis, D43776.

1.3 References

1.3.1 Document References

The CMN GEO library of functions was derived from the VIIRS SDR Geolocation module (science code) and therefore the software documents relevant to the algorithm described in this OAD are listed Table 1.

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

Table 1. Reference Documents

Document Title	Document Number/Revision	Revision Date
VIIRS Geolocation Unit Software Architecture	Y2479a Ver. 5 Rev. 5	31 Aug 2004
VIIRS Geolocation Unit Detailed Interface Control Document	Y3243 Ver. 5 Rev. 3	31 Aug 2004
VIIRS Geolocation Unit Detailed Data Dictionary	Y3248 Ver. 5 Rev. 3	31 Aug 2004
VIIRS Geolocation Unit Detailed Design	Y3245 Ver. 5 Rev. 4	31 Aug 2004
VIIRS Geolocation Algorithm Theoretical Basis Document (ATBD)	D43776 Rev. A	21 May 2008
VIIRS Geolocation SDR Science Grade Software Unit Test Document	D39301 Rev. C	21 May 2008
NPP EDR Production Report	D37005 Rev. C	16 Mar 2007
EDR Interdependency Report	D36385 Rev. E	28 Jan 2009
NPP Mission Data Format Control Book (MDFCB)	CCR Form_429--9-02-131 Rev B	14 Apr 2006
CDFCB-X Volume I - Overview	D34862-01 Rev. D	30 Jan 2009
CDFCB-X Volume II – RDR Formats	D34862-02 Rev. C	23 Jan 2009
CDFCB-X Volume III – SDR/TDR Formats	D34862-03 Rev. C	23 Jan 2009
CDFCB-X Volume IV Part 1 – IP/ARP/GEO Formats	D34862-04-01 Rev. C	23 Jan 2009
CDFCB-X Volume IV Part 2 – Atmospheric, Clouds, and Imagery EDRs	D34862-04-02 Rev. C	23 Jan 2009
CDFCB-X Volume IV Part 3 – Land and Ocean/Water EDRs	D34862-04-03 Rev. C	23 Jan 2009
CDFCB-X Volume IV Part 4 – Earth Radiation Budget EDRs	D34862-04-04 Rev. C	23 Jan 2009
CDFCB-X Volume V - Metadata	D34862-05 Rev. D	16 Jan 2009
CDFCB-X Volume VI – Ancillary Data, AuxiliaryData, Reports, and Messages	D34862-06 Rev. F	30Jan 2009
CDFCB-X Volume VII – NPOESS Downlink Formats	D34862-07 Rev. A	30 Jan 2009
NPP Command and Telemetry (C&T) Handbook	D568423 Rev. C	30 Sep 2008
D35836_G_NPOESS_Glossary	D35836_G Rev. G	10 Sep 2008
D35838_G_NPOESS_Acronyms	D35838_G Rev. G	10 Sep 2008
Data Processor Inter-Subsystem Interface Control Document (DPIS ICD)	D35850 Rev. V	11 Feb 2009
Astronomical Algorithms, Willman-Bell Inc., Richmond VA	2nd Edition	1998
USNO sers7 IERS Bulletin A	NA	NA
NGST/SE technical memo – NPP_Geo_B1.4OpsDelivery_CheckoutSummary	NP-EMD-2006.510.0073	3 Oct 2006
NGST/SE technical memo – Geo_CmnGeoMissingEphemAttitude	NP-EMD-2007.510.0023	16 Mar 2007
NGST/SE technical memo – NPP_Geo_CmnGeoMissingEphemAttitude_RevA	NP-EMD-2007.510.0023 Rev. A	31 July 2007
NGST/SE technical memo – 050808_SDRL141CmnGeoOAD_NGSTcomments	NP-EMD.2005.xxx.xxxx	8 Aug 2005
NGST/SE technical memo – EM050106Parameters-M	NP-EMD-2005.510.0004	6 Jan 2005
NGST/SE technical memo – EM050520EncInterp-M	INP-EMD-2005.510.0059	20 May 2005
NGST/SE technical memo – Eclipse Memo	NP-EMD-2005.510.0087	1 Aug 2005
NGST/SE technical memo – SAA Memo	NP-EMD.2005-510.0089	1 Aug 2005
NGST/SE technical memo – Glint Memo	NP-EMD-2005.510.0090	4 Aug 2005

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

1.3.2 Source Code References

The science and operational code and associated documentation relevant to the algorithms described in this OAD are listed in Table 2.

Table 2. Source Code References

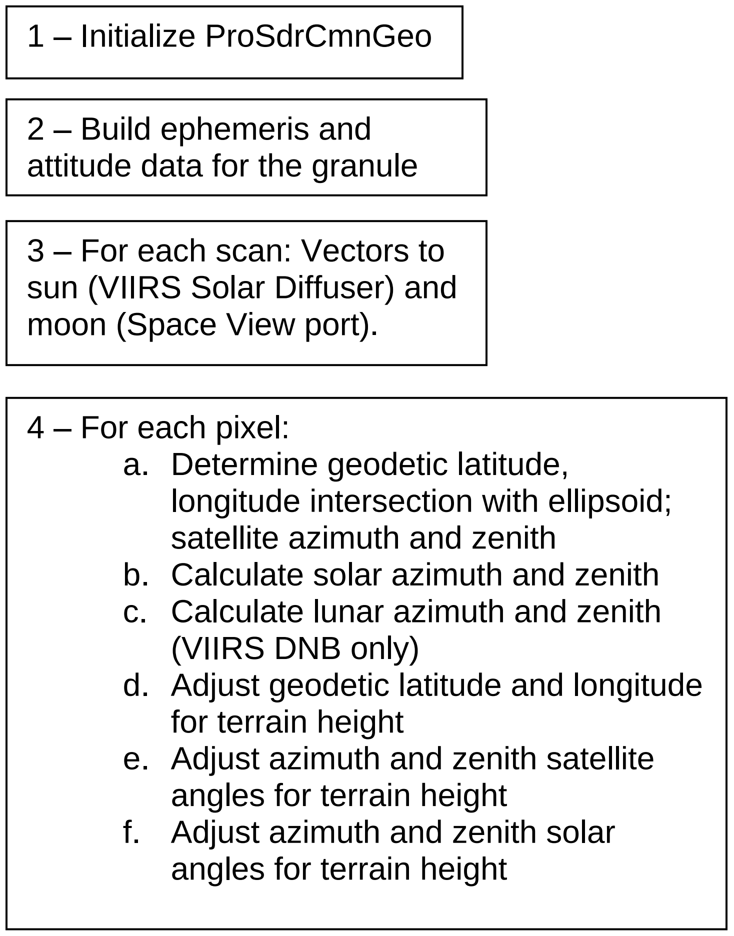
Reference Title	Reference Tag/Revision	Revision Date
Unit Test Data	N/A	N/A
VIIRS SDR GEO science-grade software (original reference source)	N/A	N/A
Common Geolocation operational software	N/A	N/A
NGST/SE technical memo – Geo_CmnGeoMissingEphemAttitude	NP-EMD-2007.510.0023	16 Mar 2007
NGST/SE technical memo – NPP_Geo_CmnGeoMissingEphemAttitude_RevA	NP-EMD-2007.510.0023 Rev. A	31 July 2007
NGST/SE technical memo – Eclipse Memo	NP-EMD-2005.510.0087	1 Aug 2005
NGST/SE technical memo – SAA Memo	NP-EMD.2005-510.0089	1 Aug 2005
NGST/SE technical memo – Glint Memo	NP-EMD-2005.510.0090	4 Aug 2005
OAD – ProSdrCmnGeo OAD Rev. A6	Build 1.5.PRO MAY	20 Jun 2007
OAD – ProSdrCmnGeo OAD Rev. A8	1.5.x.1-H (PCR018815)	30 Oct 2008
	1.5.x.1-J (PCR019133)	22 Dec 2008

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

2.0 ALGORITHM OVERVIEW

The purpose of the CMN GEO library of functions is to implement reusable functions for multiple SDR software modules. All SDR modules perform the same basic operations for geolocation (summarized in Figure 1 below). The calculation unique to the SDR is determining the view vector (aka exit vector) from the instrument. The functions common to all SDR modules make up the CMN GEO library.

CMN GEO is used to determine geodetic longitude and latitude both to the WGS84 ellipsoid and the local terrain height. Additionally, CMN GEO determines derived products such as satellite zenith and azimuth angles, solar zenith and azimuth angles, and (for the VIIRS Day/Night band (DNB)) lunar zenith and azimuth angles. CMN GEO also has routines that calculate vectors to the Sun or Moon in sensor frame coordinates and determines if the moon is in the space view or sun in the solar diffuser look. See Figure 1 for an example of the CMN GEO processing chain in an SDR algorithm. Note that steps 4d, 4e, and 4f are for Terrain Correction (described in section 2.1.2.5) and do not apply to all SDRs. Validation of spacecraft ephemeris and attitude (E&A) data is performed when the E&A packets are read and the data byte aligned. This allows the SDR module to make some decisions about graceful degradation before processing begins.



1 – Initialize ProSdrCmnGeo

2 – Build ephemeris and attitude data for the granule

3 – For each scan: Vectors to sun (VIIRS Solar Diffuser) and moon (Space View port).

4 – For each pixel:

a. Determine geodetic latitude, longitude intersection with ellipsoid; satellite azimuth and zenith
b. Calculate solar azimuth and zenith
c. Calculate lunar azimuth and zenith (VIIRS DNB only)
d. Adjust geodetic latitude and longitude for terrain height
e. Adjust azimuth and zenith satellite angles for terrain height
f. Adjust azimuth and zenith solar angles for terrain height

Figure 1. Common Geolocation Processing Chain

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)

2.1 Common Geolocation Description

2.1.1 Interfaces

Each of the CMN GEO functions is called within the SDR processing. In addition to the internal interface with the SDR processing algorithm, CMN GEO has an external (Input Only) interface with the Data Management Subsystem (DMS) to retrieve planetary ephemeris, polar wander, digital elevation model data, Two Line Element (TLE) data, and South Atlantic Anomaly (SAA) algorithm coefficients. CMN GEO does not create the Geolocation products or store them in DMS for delivery. CMN GEO uses the Naval Observatory Vector Astronomy System (NOVAS) libraries to perform some coordinate transformations and retrieve solar and lunar zenith and azimuth angles. Some minor modifications have been made to the Naval Observatory Vector Astronomy System – C version (NOVAS-C) package to support the interface. CMN GEO uses ProCmnMath for vector and matrix arithmetic and CMN GEO uses the Infrastructure (INF) Time Utility for time calculations and conversions.

2.1.1.1 Inputs

See Table 3 for the CMN GEO library inputs. Refer to the DPIS ICD, D35850, for a detailed description of the inputs acquired from DMS.

Table 3. Common Geolocation Inputs

Input	Type	Description	Units / Valid Range
View Vector	Binary	Line-of-sight vector for instrument detector (sensor frame).	Unitless / -1.0 to 1.0
Time	Float64	IET Time associated with the view vector	Micro-seconds / 1325376032000000 to 2903299232999999 (2000-01-01 00:00:00.000000Z to 2049-12-31 23:59:59.999999Z)
T_inst2sc	Float64 / (3,3)	3x3 Instrument to Spacecraft frame transformation matrix	Unitless / 3x3 Identity matrix (this signifies no sensor mounting errors)
Planetary Ephemeris	Binary	Position & velocity of celestial bodies (Sun and Moon)	Units: AU and AU/day
Ephemeris & Attitude	Binary	Time, Position, Velocity, and attitude of spacecraft	Micro-seconds / 1325376032000000 to 2903299232999999 (2000-01-01 00:00:00.000000Z to 2049-12-31 23:59:59.999999Z) , Meters / 0 to 7,200,000 , meters per second / -6,700 to + 6,700, and quaternions [unitless] / -1 to +1
Polar Wander and (UT1-UTC)	Binary	Polar wander and Earth rotation data from US Naval Observatory	Arc seconds / -6 to +6 and seconds / -1 to +1

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

Input	Type	Description	Units / Valid Range
Digital Elevation Model	Binary	Database of the earth's surface height in MSL meters, derived from SRTM30 Version 2 data (which uses USGS GTOPO30 data for areas North of 60° N and South of 60° S)	Meters / -405 to + 8752
Ellipsoid-Geoid Separation	Binary	EGM96 model of the vertical separation between the WGS84 Ellipsoid and the NIMA Geoid	Meters / -105 to +86
SAA Algorithm Coefficients	Binary	SA center (lat, lon), maximum index, latitude height and longitude width	Radians for lat/lon, Max index = estimated max events per year at center
Two Line Element	Text	14 parameters calculated and provided by C3S to IDPS describing the orbit of the spacecraft	Various / As described on the NORAD website http://www.celestrak.com/NORAD/documentation/tle-fmt.asp

2.1.1.2 Outputs

See Table 4 for a summary of all of the CMN GEO library outputs. Refer to the DPIS ICD, D35850, for a detailed description of the output products written to DMS by the respective SDR Algorithms. NOTE: The CMN GEO library does not directly output anything to DMS, so the outputs listed in Table 4 do not reflect the final Geo Product. The CMN GEO library functions return the angle outputs in radians for internal processing. These parameters are converted to degrees in the PRO COMMON code prior to outputting to DMS. These outputs are used for both Ellipsoid Intersection and optionally for Terrain Correction.

Table 4. Common Geolocation Outputs

Output	Data Type/size	Description	Fill Value
Latitude	Float32	Geodetic latitude of the Earth sample in radians.	-999.8
Longitude	Float32	Longitude of the Earth sample in radians.	-999.8
Surface Height	Float32	Surface height in meters above mean sea level (MSL)..	-999.8
Ellipsoid-geoid separation	Float32	EGM96 model of the vertical separation between the WGS84 Ellipsoid and the NIMA Geoid. Value is in meters.	-999.8
Spacecraft Zenith angle	Float32	Zenith angle of the spacecraft as viewed from the Earth sample location measured from the local normal in radians.	-999.8
Spacecraft Azimuth angle	Float32	Azimuth angle of the spacecraft as viewed from the Earth sample location measured from the local North towards East in radians.	-999.8
Solar zenith angle	Float32	Zenith angle of the Sun as viewed from Earth sample location measured from the local normal in radians.	-999.8
Solar azimuth angle	Float32	Azimuth angle of the Sun, as viewed from the Earth sample location measured from the local North towards East in radians.	-999.8

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

Output	Data Type/size	Description	Fill Value
Lunar zenith angle	Float32	Zenith angle of the Moon, as viewed from the Earth sample location measured from the local normal in radians. For VIIRS DNB only.	-999.8
Lunar azimuth angle	Float32	Azimuth angle of the Moon as viewed from the Earth sample location measured from the local North towards East in radians. For VIIRS DNB only.	-999.8
Lunar Phase	Float32	Lunar phase angle is the angle between the moon-to-sun vector and the moon-to-ground point vector, computed once per granule at the mid-granule time. This parameter is computed for VIIRS DNB only. This parameter is returned in radians.	-999.8
Moon vector	Float32	Moon vector in spacecraft frame, at time sensor is at space view port, each component in meters to Moon.	-999.8 for each component
Sun vector	Float32	Sun vector in spacecraft frame, at time VIIRS is on the Solar Diffuser, each component in meters to the Sun.	-999.8 for each component
SAA intensity	Float32	Estimated number of single event upsets per year, at a given time and location, due to increased radiation within the South Atlantic Anomaly.	-999.5
Sun glint angle	Float32	Cosine of the sun glint angle, i.e., the angle between the viewing direction and the direction into which sunlight would be reflected by a specularly reflecting, flat, horizontal, surface.	-999.5
Lunar Eclipse Status	bool	True if a lunar eclipse is occurring at a given time.	false
Solar Eclipse Status	bool	True if a solar eclipse is occurring at a given time and location.	false

2.1.2 Algorithm Processing

The following is an overview of functionality implemented by the CMN GEO library of functions. The functions and methods are divided into several groups and each function is described in more detail in the following subsections. Some functions are used internally to Cmn Geo to perform various calculations, while others are provided as an interface for an algorithm to use the Cmn Geo library. These interface functions are labeled ‘public’ in the detailed descriptions provided below.

1. The Initialization and Termination Functions in section 2.1.2.1 are used to create an instance to use when calling the Cmn Geo library and for cleaning up the class after it has been used. These methods retrieve any inputs needed by Cmn Geo and provide access to the library of functions.
2. The Functions for Ephemeris and Attitude Processing in section 2.1.2.2 are used for processing the ephemeris and attitude data so that Cmn Geo can use it to calculate spacecraft position. These functions detect and fill any possible gaps in the attitude and ephemeris data and to calculate the sub satellite point on the earth. Most of them are used internally to Cmn Geo, they may retrieve inputs from DMS, if needed, and do not have external interfaces to be used by an algorithm.
3. The Functions for Coordinate Transformation and Ellipsoid Intersection in section 2.1.2.3 are used for calculating the ellipsoid intersection and performing various coordinate transformations. Most of these functions are only used internally by Cmn Geo.

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

4. The Functions for Retrieving Solar and Lunar Geometry in section 2.1.2.4 are used for determining angles to the sun or moon relative to the spacecraft or the point on the ground. This group also includes functions for checking for sun glint, the presence of solar or lunar eclipse, and whether the sun or moon are in the view of the solar diffuser or the spaceview. The sunAngles and moonAngles methods can be used to adjust the specified angles for Terrain Correction after having determined the terrain corrected latitudes and longitudes.
5. A detailed description of Adjusting Geolocation for Terrain is given in section 2.1.2.5 and the functions used for performing Terrain Correction are listed in section 2.1.2.6.
6. The Functions for South Atlantic Anomaly in section 2.1.2.7 are used for estimating the likelihood of increased radiation within the South Atlantic Anomaly (SAA).

2.1.2.1 Initialization/Termination Functions of the ProSdrCmnGeo Singleton Class

2.1.2.1.1 getInstance() - public

This function returns a pointer to an instance of the ProSdrCmnGeo class. If an instance needs to be created, it is created. The function initCmnGeo() is called to initialize this new instance.

2.1.2.1.2 clear() : void - public

This function is called by the SDR process to clear the ephatt_ structure. This function must be called before calling setupEphatt().

2.1.2.1.3 initCmnGeo()

This initializes ProSdrCmnGeo with static ancillary data from DMS and initializes structures used by NOVAS-C. The static ancillary data used by NOVAS-C is the JPL Planetary Ephemeris and the Polar Wander / UT1-UTC data from USNO sers7 IERS Bulletin A.

2.1.2.1.4 getDMSClient()

This function returns a pointer to the DMS client that CMN GEO is using.

2.1.2.1.5 getShortName()

This function is called to retrieve the Common Short Name’s (CSN) for the polar wander and planetary ephemeris data.

2.1.2.2 Functions for Ephemeris and Attitude Processing

2.1.2.2.1 setupEphAtt() - public

This function reads the spacecraft E&A byte-aligned Raw Data Records (RDR) for the start/stop time of the granule and places the data into the ephemeris and attitude structure (ephatt).

2.1.2.2.2 satPosAtt() - public

Given the observation time, this function finds a point in the ephatt_ points map field for the specified time. The point found is used to determine the location on the earth.

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

2.1.2.2.3 fillEphAttPoint()

Given the byte-aligned ephemeris and attitude data, this function fills one point of information in the map of points in the ephatt_ structure. (It calculates the geodetic latitude, geocentric latitude, longitude, etc.)

2.1.2.2.4 initPointsMap()

This function initializes the points map in the ephatt_structure.

2.1.2.2.5 buildQuatMatrix()

This function builds the quaternion matrix.

2.1.2.2.6 getEAPointMap()

This function returns the ephemeris and attitude data in a Standard Template Library(STL) map object.

2.1.2.2.7 buildECIOrbFrame()

This function uses the ECI position and velocity to build the Orbit Frame matrix.

2.1.2.2.8 detectAndFillGaps()

This function determines if there are any missing S/C Ephemeris & Attitude data points and fill in those missing points to create E&A data sampled at one second intervals.

2.1.2.2.9 quadraticInterpolation()

This function performs Quadratic Interpolation of the three input points for the time specified in the output point.

2.1.2.2.10 sgp4_predEA()

This function calculates all of the ephemeris point data, for the satellite in the orbElem_ structure at the requested time.

2.1.2.2.11 sgp4_rtrvTLE()

This function retrieves the Two Line Element (TLE) strings from DMS and puts them into the Orbital Elements member variable.

2.1.2.2.12 sgp4_TLE()

This function converts TLE strings in the orbElem_ struct to the numbers needed by the SGP4 ephemeris prediction model. The output numbers are stored in orbElem_.

2.1.2.2.13 sgp4_calc()

Acts as an interface between other programs and the SGP4 ephemeris prediction model Fortran code.

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)

2.1.2.2.14 actan()

Fortran code to calculate an arctangent. This code was provided with the SGP4 ephemeris prediction model Fortran code.

2.1.2.2.15 fmod2p()

Fortran code to force the input value into the range 0 to 2 PI. This code was provided with the SGP4 ephemeris prediction model Fortran code.

2.1.2.2.16 sgp4()

Returns space craft coordinates and velocity for the current time value.

2.1.2.2.17 sgp4_driver()

Fortran code which acts as the driver for the SGP4 model. This code is called from a C program to interface with the SGP4 model.

2.1.2.3 Functions for Coordinate Transformation and Ellipsoid Intersection

2.1.2.3.1 calcGDRollPitchYaw()

This function transforms the ECI position, velocity, and quaternions to roll, pitch, and yaw in the ECI system.

2.1.2.3.2 eciRPYtoQuat()

This function uses the orbit frame and the ECI roll, pitch, and yaw to calculate quaternions.

2.1.2.3.3 ellipIntersect()   - public

Given the sensor exit vector and the ephemeris and attitude, this function calculates the geodetic latitude, longitude, and satellite azimuth and zenith angles of a pixel, for the point where the LOS intersects the WGS84 Ellipsoid. The steps performed by this method are:

1) Convert a single pixel's exit vector Line-of-Sight (LOS) to Spacecraft (S/C) coordinates using a transformation matrix that reflects the actual instrument mounting and sensor to spacecraft transformation.
2) Use the quaternion matrix to rotate the corrected exit vector from S/C to ECI. This is a simple matrix multiplication performed by matrixVectorProduct().
3) Convert the exit vector from ECI to ECR. This step is performed by the convCoordSys() method using the input flag of ECI2ECR.
4) Intersect each pixel's line of sight with the WGS84 ellipsoid and output geodetic latitude, longitude.
5) Compute derived ellipsoid geolocation products, such as solar and sensor azimuth and zenith angles, sensor range, and lunar azimuth and zenith angles. Note: lunar azimuth and zenith angles are used for VIIRS DNB only.

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

2.1.2.3.4 calcSatAzmZen() – public

This function calculates the satellite azimuth and zenith angles based on the input latitude, longitude, and sample-to-satellite vector.

2.1.2.3.5 validLat()

This function returns false if the value of latitude in radians is less than –PI/2 or greater than PI/2; otherwise returns true.

2.1.2.3.6 validLon()

This function returns false if the value of longitude in radians is less than –PI or greater than PI; otherwise returns true.

2.1.2.3.7 getPolarUT1UTC()

Given the IET observation time, this function finds the polar wander and Universal Time One (UT1) minus Universal Coordinated Time (UTC) data (the difference between UT1 and UTC reflects the Earth's rotation).

2.1.2.3.8 convCoordSys()

Given IET observation time, position, and velocity, this function converts position and velocity in ECR or ECI coordinates to ECI or ECR coordinates based on the direction specified by the convFlag. Note: GPS uses the terms ECEF and ECSF. ECEF=ECR and ECSF=ECI.

This method uses the invPNSW() and invWobble() functions described below when converting from ECI to ECR. When converting from ECR to ECI, the non-inverse functions provided in NOVAS-C are used. Note that there are no inverse functions implemented for the NOVAS-C functions precession(), nutate(), or spin() because they have an inverse capability built into them already.

2.1.2.3.9 tdb2tdt()

NOVAS-C function is used for time conversion from Solar System Barycentric time to Terrestrial Dynamic time.

2.1.2.3.10 earthtilt()

NOVAS-C function is used for changes of obliquity and application of the equation of equinoxes.

2.1.2.3.11 sideral_time()

NOVAS-C function is used to convert rotation of the earth to Greenwich hour angle.

2.1.2.3.12 convVec2LatLon()

This converts the position vector into geodetic latitude, geocentric latitude, and longitude.

2.1.2.3.13 ecr2Eci() - public

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

This function converts position and velocity vectors from ECR to ECI coordinates. This function calls convCoordSys() to do the conversion.

2.1.2.3.14 eci2Ecr() - public

This function converts position and velocity vectors from ECI to ECR coordinates. The function calls convCoordSys() to do the conversion.

2.1.2.3.15 eciRPYtoQuat()

This function converts roll, pitch, and yaw to the four quaternion components.

2.1.2.3.16 invPNSW()

This function performs the inverse of the NOVAS-C function pnsw(). It is used by the convCoordSys() method to convert from ECI coordinates to ECR coordinates.

The NOVAS-C function pnsw() transforms a vector from an Earth-fixed geographic system to a space-fixed system based on mean equator and equinox of J2000.0 while applying rotations for wobble, spin, nutation, and precession. The inverse function performs the inverse calculations, but in reverse order.

2.1.2.3.17 invWobble()

This function performs the inverse of the NOVAS-C function wobble(). It is used by the convCoordSys() method to convert from ECI coordinates to ECR coordinates.

The NOVAS-C function wobble() corrects Earth-fixed geocentric rectangular coordinates for polar motion. It transforms a vector from Earth-fixed geographic system to a rotating system based on rotational equator and orthogonal Greenwich meridian through axis of rotation. The inverse function performs the rotation in the opposite direction.

2.1.2.4 Functions for Retrieving Solar and Lunar Geometry

2.1.2.4.1 sunAngles() - public

Given the IET observation time, geodetic latitude and longitude in radians, this function calculates the azimuth and zenith angles in radians of the Sun. This function calls sunMoonAngles() to calculate these values.

2.1.2.4.2 vectorAtSat() - public

Given the interpolated ephemeris and attitude data and flag specifying sun or moon, this function calculates the x, y, z vector to the Sun or Moon in spacecraft frame at the location of the satellite. Units are in meters.

2.1.2.4.3 moonInView() - public

Given the vector to the moon (SC frame), sensor to SC rotation matrix, and sensor mounting error matrix, this function is used to determine whether the moon is in the spaceview. This function calls satPosAtt(), vectorAtSat(), and objectInView().

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

2.1.2.4.4 sunInView() - public

Given the vector to the sun (SC frame), sensor to SC rotation matrix, sensor mounting error matrix, this function is used to determine whether the sun is in the solar diffuser. This function calls satPosAtt(), vectorAtSat(), and objectInView().

2.1.2.4.5 moonAngles() - public

Given the IET observation time, geodetic latitude and longitude in radians, this function calculates the azimuth and zenith angles of the Moon in radians. It also calculates the Moon phase in radians and the Moon illumination fraction. This function calls the sunMoonAngles() to calculate these values.

2.1.2.4.6 sunMoonAngles() - public

This function is called by sunAngles() and moonAngles() to calculate the values described in those functions.

2.1.2.4.7 objectInView()

This function calculates the angle between a vector in spacecraft coordinates to an object (Sun or Moon) and an exit vector from a sensor (solar diffuser or spaceview port).

2.1.2.4.8 ascendDescendIndicator() - public

This function determines whether a granule is ascending or descending for a specific time.

2.1.2.4.9 vectorAtGeoCenter()

This function calculates a vector from the center of the earth to either the Sun or the Moon. The vector components are in meters and the coordinate system is ECI, J2000.

2.1.2.4.10 vectorAtOrbitFrame()

This function calculates a vector in the orbit frame from the S/C to either the Sun or the Moon. The vector components are in meters.

2.1.2.4.11 calcSunGlintAngle()

Given a satellite's azimuth and zenith angles and solar azimuth and zenith angles, this function computes the sun glint angle, i.e., the angle between the viewing direction and the direction into which sunlight would be reflected by a specularly flat, horizontal, surface. Note that for efficiency, the value returned is actually the cosine of the angle and not the angle itself.

Inputs:

1. inSatAzm - Input satellite azimuth angle in radians
2. inSatZen - Input satellite zenith angle in radians
3. inSunAzm - Input sun azimuth in radians
4. inSunZen - Input sun zenith in radians

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

Outputs:

1. outCosReflAngle - Cosine of the reflected sun angle.
2. return status - PRO_SUCCESS or an error code

As per Tech Memo, Source of Glint Data, 2005/08/04 (NP-EMD.2005.510.0090), the formula for sun glint is that used by the VIIRS Cloud Mask algorithm. Note that it is the cosine of the angle that is returned and not the angle itself. The calculations below are identical to the IDPS implementation of the Sun Glint calculation in the ProSdrCmnGeo::calcSunGlintAngle() method and to the implementation of the Sun Glint calculation in the VIIRS CloudMask sun_glint() method. Also note that both of these implementations are mathematically identical to, but more efficient than, the calculation in the Tech Memo. See the derivation below.

```
zenCosMinus = cos(inSunZen – inSatZen)
zenCosPlus = cos(inSunZen + inSatZen)
outCosReflAngle =  0.5 * ((zenCosMinus + zenCosPlus) +
                          ((zenCosMinus - zenCosPlus) *
                            cos(PI - (inSatAzm - inSunAzm))));
```

The zenith angle is measured from the vertical to the line-of-sight vector to the satellite or solar position; the azimuth angle is measured clockwise from north. At the poles, zero azimuth is along the Prime Meridian. Figure 2 below illustrates the definition of the angles.

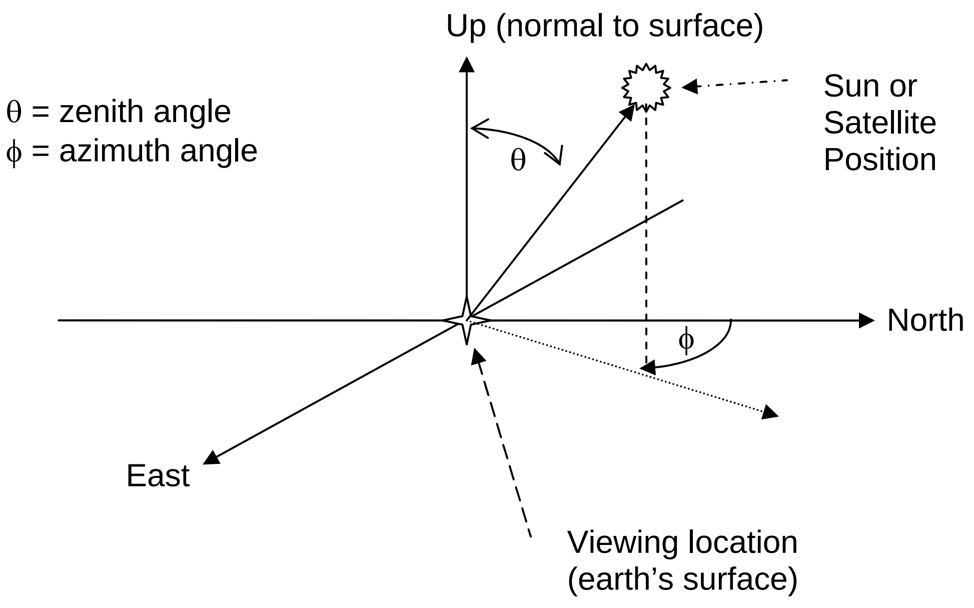


Figure 2. Azimuth – Zenith Coordinates

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

2.1.2.4.12 Sun Glint Angle derivation

The equation in the Tech Memo, Source of Glint Data, 2005/08/04 (NP-EMD.2005.510.0090), is as follows (using the angle names from the Cmn Geo code):

cos(ReflAngle) = sin(inSatZen)*sin(inSunZen) *
cos(180 – (inSunAzm – inSatAzm)) +
cos(inSatZen) * cos(inSunZen); (1)

By Trigonometric identity:

cos(A)*cos(B) = ½*cos(A-B) + ½*cos(A+B) (2)
sin(A) * sin(B) = ½*cos(A-B) - ½*cos(A+B) (3)

Substituting (2) & (3) into (1) gives:

cos(ReflAngle) =
[½*cos(inSatZen - inSunZen) - ½*cos(inSatZen + inSunZen)] *
cos(180 – (inSunAzm – inSatAzm)) +
[½*cos(inSatZen - inSunZen) + ½*cos(inSatZen + inSunZen)] (4)

Simplifying (4) gives:

cos(ReflAngle) =
½ * [(cos(inSatZen - inSunZen) - cos(inSatZen + inSunZen)) *
cos(180 – (inSunAzm – inSatAzm)) +
(cos(inSatZen - inSunZen) + cos(inSatZen + inSunZen))] (5)

Using the following definitions and substituting into (5) gives:

zenCosMinus = cos(inSatZen – inSunZen) (6)
zenCosPlus = cos(inSatZen + inSunZen) (7)
cos(ReflAngle) =
½ * [(zenCosMinus - zenCosPlus) *
cos(180 – (inSunAzm – inSatAzm)) +
(zenCosMinus + zenCosPlus)] (8)

Equation (8) can be reordered as:

cos(ReflAngle) =

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

½ * [(zenCosMinus + zenCosPlus) +
(zenCosMinus - zenCosPlus) *
cos(180 – (inSunAzm – inSatAzm))] (9)

The Cmn Geo code uses the following definitions as copied from the previous section 2.1.2.4.11:

zenCosMinus = cos(inSunZen – inSatZen) (10)

zenCosPlus = cos(inSunZen + inSatZen) (11)

Rewriting Equations (6) and (7) by reordering the angles gives:

zenCosMinus = cos(– (inSunZen – inSatZen)) (12)

zenCosPlus = cos(inSunZen + inSatZen) (13)

By the trigonometric identity, **cos(A) = cos(-A)**, Equation (12) is identical to Equation (10) and by the commutative property, Equation (11) is identical to Equation (13). Finally, note that there is a difference in the order of the angles in the cosine term in Equation (9) and the equation implemented by the Cmn Geo code. Writing only these two cosine terms for clarification:

cos(180 – (inSunAzm – inSatAzm)) (14 – from Tech Memo)

cos(180 – (inSatAzm – inSunAzm)) (15 – From Cmn Geo)

Rewriting Equation (14) gives:

cos(180 – (– inSatAzm + inSunAzm)) or

cos(180 + (inSatAzm – inSunAzm)) (16)

By using the two trigonometric identities:

cos(180 – A) = – cos(A), and

cos(180 + A) = – cos(A)

Equation (16) is identical to Equation (15) used by the Cmn Geo code and rewriting Equation (9) using this identity becomes:

cos(ReflAngle) =

½ * [(zenCosMinus + zenCosPlus) +
(zenCosMinus - zenCosPlus) *
cos(180 – (inSatAzm – inSunAzm))] (17)

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

Therefore, the definitions defined in the Tech Memo, Source of Glint Data, 2005/08/04 (NP-EMD.2005.510.0090), using Equation (1) can be rewritten as Equation (17) as it is used in the Common Geolocation and VIIRS Cloud Mask code as they are mathematically identical. Furthermore, the computational benefit to using the Equation (17) over Equation (1) is that only three cosines are computed as opposed to three cosines and two sines.

2.1.2.4.13 checkLunarEclipse()

Check if a lunar eclipse is occurring at the given time according to the algorithms of Jean Meeus[1] and return true if one is occurring or false if not. A summary of the basic algorithm steps are as follows:

- Check if Moon phase and eclipse data has been cached and if so, check if the data spans the requested time.
- If the data does not span the requested time, then perform Moon phase calculations to determine the time of the nearest three Full Moons.
    - Perform shadow calculations to determine if the Moon is in eclipse at the time of each Full Moon.
    - Cache the phase and associated eclipse data for the three lunar cycles (this eliminates the need for recalculation until input times change by 2-4 weeks).
- Check if the nearest Full Moon is in eclipse, and if so, if the period of eclipse spans the requested time. If it does, then a lunar eclipse occurring.

For efficiency reasons, this function does not check local conditions, i.e., it does not check visibility of the Moon at a given ground location. If the Moon is in eclipse at the requested time, the invoking algorithms need to verify the Moon is visible at the scan/pixel location. Since the invoking algorithms already have Moon zenith angles available, it is more efficient for them to check the angles than for this method to compute new zenith angles.

2.1.2.4.14 checkSolarEclipseInitial()

Check if a solar eclipse is occurring at the given time according to the algorithms of Jean Meeus[1] and return true if one is occurring or false if not. A summary of the basic algorithms steps are:

- Check if Moon phase and eclipse data has been cached and if so, check if the data spans the requested time.
- If the data does not span the requested time, then perform Moon phase calculations to determine the time of the nearest three New Moons.
    - Perform shadow calculations to determine if the Sun is in eclipse at the time of each New Moon.
    - Cache the phase and associated eclipse data for the three lunar cycles (this eliminates the need for recalculation until input times change by 2-4 weeks).
- Check if the nearest New Moon is in eclipse, and if so, if the period of eclipse spans the requested time. If it does, then a solar eclipse occurring.

[1] Meeus, Jean, "Astronomical Algorithms, 2nd Edition," Willman-Bell Inc., Richmond VA, 1998, 477 pp. Chapter 54, Eclipses, pages 379-388 describes the eclipse algorithms and includes calculations from Chapter 49, Phases of the Moon, pages 349-350.

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

Note that this function does not perform detailed checks of local conditions. If this function returns true, and the sun is above the horizon at the desired location, then checkSolarEclipseFinal should be called to verify eclipse at that location.

2.1.2.4.15 checkSolarEclipseFinal()

Check if a solar eclipse is occurring at the given time and ground location, using detailed NOVAS routines. A summary of the basic algorithms steps are:

- Use NOVAS routines to get apparent position vectors of the Sun and Moon at a test location and time.
- Using the radius of the Sun and Moon and the distance to each, calculate the apparent disk size of both. The sum of the two half-disk sizes becomes a test angle.
- Use the vector dot product to determine the actual angle between the Sun and Moon vectors.
- If the actual angle is less than the test angle, then the Solar Eclipse flag is true.

For efficiency reasons, it is assumed that the invoking algorithm has verified the Sun is visible at the given ground location prior to calling this routine; checkSolarEclipseFinal does not re-check visibility of the Sun.

2.1.2.5 Adjusting Geolocation for Terrain

This section describes the implementation for adjusting the geolocation for terrain height. The method employed is slightly different than that described in the ATBD, but starting with the geodetic latitude and longitude along with satellite zenith angle and azimuth angle, the geolocation is adjusted for terrain through application of the following steps.

1) The TERECO DB is accessed to obtain the Mean Sea Level (MSL) surface height (SRTM30 Version 2) and Ellipsoid-Geoid Separation (EGM96) at the input point. The surface height above the Ellipsoid is calculated by adding those two together.

    a) If the surface height at the ellipsoid intersection, and the satellite zenith angle, combine to indicate that the potential horizontal correction is less than N meters (N is a configuration data item currently set to 35 meters), then we copy the ellipsoid intersection location to the terrain corrected geolocation object and skip to step 6.

    b) If the maximum and minimum surface height both equal 0 this is an ocean pixel. Determine the point where the LOS emerges from the ocean and correct the latitude and longitude for this location.

    c) The starting LOS point is initially set to the ellipsoid intersection point. If the Earth surface is below the ellipsoid, then we move the starting LOS point to an altitude of -600 meters and the starting distance angle is calculated. This is the angle measured at the center of the Earth from the ellipsoid intersection point to the starting LOS point (farthest from satellite). The following equation is used to calculate this negative distance angle:

    dstang_l = **elos_hgt** * (tan (**zenith**)) / **earth_radius**;

    where:

    dstang_l is the distance angle measured at the center of the Earth, from the input point to the LOS starting point farthest from the satellite

    elos_height is maximum height 32km around this point in the TERECO DB

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

zenith is the input spacecraft zenith angle, determined from Ellipsoid Intersection angle

earth_radius is the radius of the Earth at the input latitude (i.e.latitude of Ellipsoid Intersection point)

d) The ending LOS point is calculated using the maximum height 32km around the ellipsoid intersection point in the TERECO DB. The equation above is used to estimate the distance to the end of the LOS closest to the satellite. This produces a positive distance angle called dstang_h. This is the distance angle measured at the center of the Earth, from the ellipsoid intersection point to the ending LOS point (closest to satellite).

e) The number of points needed to divide the LOS into 'pieces' that are horizontally 500 meters apart is determined by:

Npts = (dstang_h - dstang_l) / (500 / earth_radius);

At the end of Step 1, the LOS point closest to the satellite (the end of the LOS) is guaranteed to be above the Earth surface, and the point farthest from the satellite (the start of the LOS) is guaranteed to be below the Earth surface. We also have the horizontal distance from the Ellipsoid Intersection to the start (which may be zero) and the end of the LOS.

2) Use function target_point, and the azimuth towards the satellite, to calculate the latitude and longitude of the start and end of the LOS. We convert the latitude and longitude of the start and end points to positions on the TERECO DB grid.

Spherical trigonometry (with earth radius determined from the latitude of the input point) is used. The equations used are the Law of Sines for Plane Oblique Spherical Triangles, and the Law of Cosines for Plane Oblique Spherical Triangles. The three points in the spherical triangle are: the North Pole, the start point, and the end point. Distance is specified as a distance measured at the center of the Earth and is therefore a distance angle rather than distance along the surface of the Earth.

The worst case distance from the Ellipsoid Intersection point to the Earth surface emergence point is about 30 kilometers along the LOS. The variation between a sphere and the WGS 84 Ellipsoid over this small distance is insufficient to change the answer by three meters. That worst case rapidly changes to less than one meter only 100 pixels from the edge of scan.

3) We now start at the LOS point closest to the satellite and calculate the next LOS point away from the satellite. Use ancillary DB access functions to obtain the MSL height, the Ellipsoid-Geoid separation, and the Earth surface height of the Ellipsoid at each point (which is the sum of those two numbers for this point). The MSL height for each point is determined by bilinear interpolation of the GTOPO30 data. The Ellipsoid Geoid Separation (EGS) numbers are a simple average of the four surrounding values. The EGM96 data is so smooth that the simple average is never more than 25 centimeters from a value determined by bilinear interpolation.

We also calculate the LOS altitude for this point. The LOS altitude is adjusted for the curvature of the Ellipsoid along the LOS. The LOS altitudes are meters above the WGS84 Ellipsoid.

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)

4) At this point we compare the LOS altitude to the Earth surface height. The search stops when the LOS altitude of the point is below the terrain. Now, the terrain intersection is found, and the next point towards the satellite is above the terrain. If the LOS altitude is not below the terrain, repeat steps 3 and 4 using the newly calculated point as the starting point.

5) Interpolate data between the two points identified in Step 4 to find the location and MSL height where the LOS emerges from the Earth surface. Since the interpolated position is in TERECO DB coordinates, we convert to latitude and longitude.

   a) By drawing a straight line between the two LOS points, and a straight line between the two Earth surface points, the fraction of a step along the LOS is determined by solving the equation for the intersection of two lines. There is no divisor in the slope equations because the two points are one LOS step apart, and the calculated result is a fraction of that one LOS step.

   ```
   ees_slope = ees_height[sidx_p1] - ees_hgt[sidx];

   los_slope = los_hgt[sidx_p1] - los_hgt[sidx];

   frac =  ( los_hgt[sidx] - ees_hgt[sidx] )  /  ( ees_slope - los_slope );
   ```

   where:

   ees-hgt is the Ellipsoid to Earth surface height, and that is the sum of the MSL height from the GTOPO30 data and Ellipsoid-Geoid separation from the EGM96 data

   los_hgt is the height of the LOS above the Ellipsoid

   sidx is the index of the point closest to the satellite, where the Earth's surface is above the LOS

   sidx_p1 is the point next to sidx, and the Earth surface at this point is checked to make certain it is below the LOS at this point

   b) Then frac is used to calculate the location where the LOS emerges from the Earth surface.

   ```
   lat_out =  lat[sidx] + { frac * (lat[sidx_p1] - lat[sidx] ) };
   ```

   An analogous equation is used for longitude, except that two points spanning the 180 degrees longitude line require variations on the above equation.

6) Complete the results by adjusting Sun, satellite, and/or Moon angles for the new location. This is achieved by calling the sunAngles and/or moonAngles methods of Cmn Geo with the terrain corrected latitude and longitude. If steps 2 thru 5 were skipped, then this data is copied from the data for the Ellipsoid Intersection. See Figure 3.

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

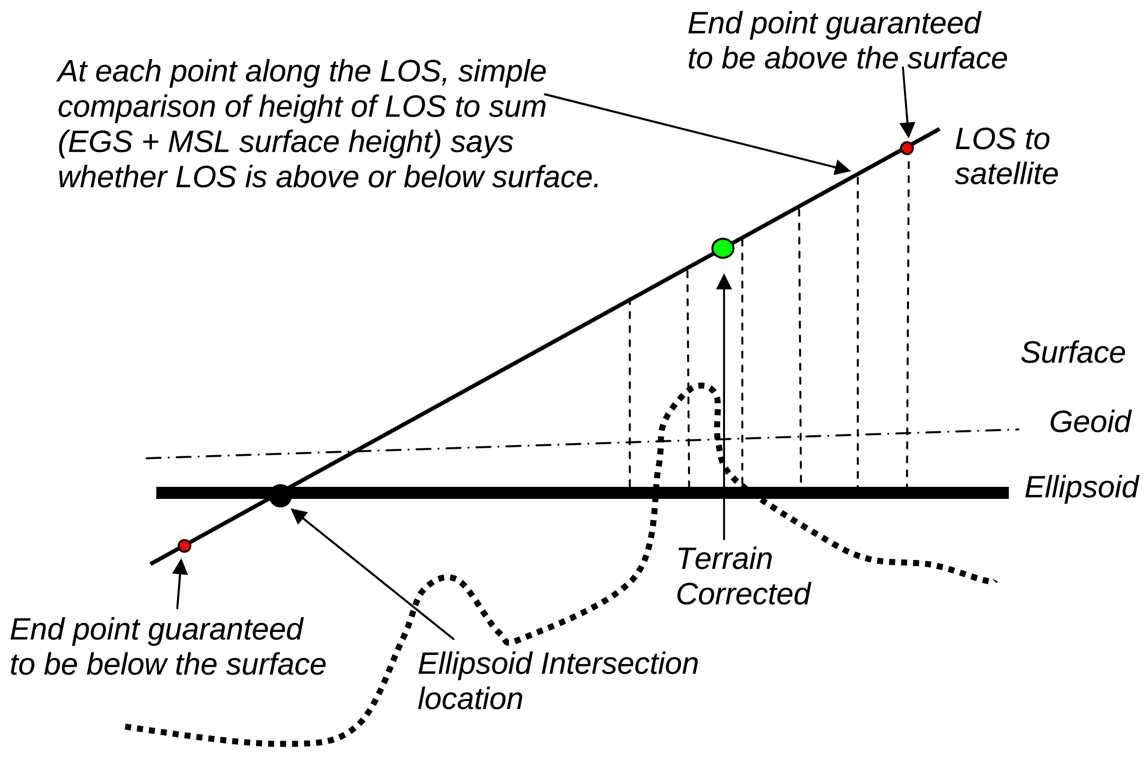

End point guaranteed to be above the surface
At each point along the LOS, simple comparison of height of LOS to sum (EGS + MSL surface height) says whether LOS is above or below surface.
LOS to satellite
Surface
Geoid
Ellipsoid
Terrain Corrected
End point guaranteed to be below the surface
Ellipsoid Intersection location

Figure 3. Terrain Adjusted Geolocation Schematic

2.1.2.6 Functions for Adjusting Geolocation for Terrain Height

2.1.2.6.1 terrainCorrection()

This is the main function for terrain correction. This function terrain corrects earth sample locations and satellite azimuth and zenith angles. The inputs are the Ellipsoid Intersection data, and the outputs are corrected to the place where the LOS emerges from the Earth's surface. When the Earth surface is above the Ellipsoid, the corrected point is towards the satellite (majority of cases), and when the Earth surface is below the Ellipsoid (Indian Ocean, Dead Sea, Death Valley, etc.) the corrected point is away from the satellite.

2.1.2.6.2 applyOceanCorrection()

This function adjusts the location of the pixel by the EGS value. This adjustment happens when the pixel is an ocean pixel that is at least 28 km from land.

2.1.2.6.3 calcTCSatAzmZen()

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

This function calculates new satellite azimuth and zenith angles using the terrain corrected latitude and longitude values, MSL height, EGS height, and ECR satellite position vector.

2.1.2.6.4 setupLOSPoints()

This function creates the starting LOS point (farthest from S/C) and ending LOS point (closest to S/C). It then determines if the terrain intersection is above the end point or below the end point. (Based on the maxHgt field, most cases the terrain intersection is below the end point).

2.1.2.6.5 calcGridFromLatLon()

This function calculates new latitude/longitude values for an LOS point and then uses the latitude/longitude values to get terrain DB grid coordinates.

2.1.2.6.6 fillTerrainPoint()

This function retrieves a Digital Elevation Model (DEM) tile from DMS if necessary and fills an LOS point with the MSL, EGS (Ellipsoid – Geoid Separation), and EES (Ellipsoid to Earth Surface) height information.

2.1.2.6.7 findTerrain()

This function calculates a terrain corrected latitude/longitude, the MSL surface height, and satellite azimuth/zenith angles.

2.1.2.6.8 fillTerrainPointBiInterp()

This function fills an LOS point with MSL, EGS, EES, maxHgt, minHgt information using bi-linear interpolation. The four corner points around the point of interest are used in the interpolation.

2.1.2.6.9 calcLOSHgt()

This function calculates the LOS height of a point.

2.1.2.6.10 calcStartPoint()

This function fills the LOS point farthest away from the spacecraft with MSL, EGS, EES, maxHgt, minHgt information. It also calculates if the point is below the terrain and moves the point far enough away to ensure it is below all terrain.

2.1.2.6.11 determineIntersectAboveEndPt()

This function is called if the end point calculated using the maxHgt is below the terrain. This function moves along the LOS towards the spacecraft until the terrain intersection is found.

2.1.2.6.12 calcAndFillPoint()

This function calculates the latitude/longitude of an LOS point and also fills the LOS point with MSL, EGS, EES, maxHgt, minHgt information.

2.1.2.6.13 determineIntersectBelowEndPt()

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

This function is called if the end point calculated using the maxHgt is above the terrain. This function moves along the LOS away from the spacecraft until the terrain intersection if found.

2.1.2.7 Functions for South Atlantic Anomaly

2.1.2.7.1 getSAAIntensity()

For a given location, getSAAIntensity estimates the number of single event upsets per year. A higher number indicates a greater danger of increased radiation within the South Atlantic Anomaly (SAA) causing earth location errors within optical encoders and false hits within detector arrays.

Inputs:

1. inLat - Latitude of location in radians
2. inLon - Longitude of the location in radians

Outputs:

1. outIntensity - Estimated event index
2. return status - PRO_SUCCESS or an error code

As per Tech Memo, Source of South Atlantic Anomaly Data, 2005/08/01 (NP-EMD.2005.510.0089), the South Atlantic Anomaly is modeled as a 5-parameter Gaussian distribution parallel to the lat/lon lines, in accordance with an unpublished paper by the Toronto Mopitt Team, "The South Atlantic Anomaly seen by MOPITT Instrument," DRAFTsaa, 22Nov2002. The equation used to model the anomaly is

$$S_{index} = I_{\max} e^{-\left\{ \frac{\left(\frac{\left| \phi_{in} - \phi_{center} \right|}{\phi_{height}} \right)^2 + \left(\frac{\left| \lambda_{in} - \lambda_{center} \right|}{\lambda_{width}} \right)^2}{2\sigma^2} \right\}}$$

where S_{index} = output SAA index value,

I_{max} = maximum output value,

ϕ_{in} = input latitude,

ϕ_{center} = *l*atitude at center of SAA ,

ϕ_{height} = latitude height, of one standard deviation,

λ_{in} = input longitude,

λ_{center} = longitude at center of SAA,

λ_{width} = longitude width, of one standard deviation, and $2\sigma^2$ = 1.0.

The values for I_{max}, ϕ_{center}, ϕ_{height}, λ_{center}, and λ_{width}, are all obtained from the CmnGeo-SAA-AC-Int algorithm coefficients table.

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)

2.1.3 Graceful Degradation

2.1.3.1 Graceful Degradation Inputs

None.

2.1.3.2 Graceful Degradation Processing

None.

2.1.3.3 Graceful Degradation Outputs

None.

2.1.4 Exception Handling

The CMN GEO module produces two error messages— PRO_FAIL and PRO_GEO_ WARNING.

PRO_FAIL is issued under the following conditions:

1. Cannot retrieve data from DMS.
2. Internal pointers are 0 (aka NULL).
3. Cannot initialize NOVAS-C structures.
4. INF Time Utility throws an exception.
5. Specified time is out of range for JPL Planetary Ephemeris data in NOVAS-C functions.

The methods in CMN GEO returns a PRO_FAIL if any of the above conditions are encountered.

PRO_GEO_WARNING is issued under the following conditions:

1. Input vector is 0 (aka NULL).
2. E&A Point tai field is equal to FILL data.
3. Input view vector is a NaNQ.
4. The view vector does not intersect the ellipsoid.
5. The latitude or longitude value that is calculated or used as input is not a valid value.
6. An input time is out of range in the S/C E&A RDR data.
7. The ProSdrCmnGeo has not been properly cleared before using.

The methods in CMN GEO returns PRO_GEO_WARNING if any of the above conditions are encountered.

NOTE: The detectAndFillGaps() method returns PRO_GEO_SHORTCUT, if it detects and fills a Stage 3 type gap. This return value is caught by the initPointsMap() method and converted to a PRO_SUCCESS to designate a successful completion of the Common Geolocation Anomaly processing.

2.1.5 Data Quality Monitoring

See the SDR OADs for each sensor (i.e., VIIRS, OMPS, CrIS, ATMS, etc.).

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

2.1.6 Computational Precision Requirements

All internal calculations done at double precision as small scan angle errors lead to large geolocation errors. The geodetic Latitude and Longitude numbers are stored in the geolocation objects as 32-bit floats, which can introduce a computational precision no larger than two meters in Earth location.

2.1.7 Algorithm Support Considerations

INF and DMS must be running before a Common Geo instance is used.

2.1.8 Assumptions and Limitations

2.1.8.1 Assumptions

The CMN GEO algorithm assumes prior to processing, that the proper number of S/C E&A RDRs has been retrieved and byte-aligned. If the proper number of S/C E&A RDRs could not be retrieved, then the CMN GEO code attempts to retrieve the Two-Line Element data and calculate the missing Ephemeris and Attitude data.

2.1.8.2 Limitations

The Terrain Correction algorithm has not been approved by the ACCB.

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

3.0 GLOSSARY/ACRONYM LIST

3.1 Glossary

The current glossary for the NPOESS program, D35836_E_NPOESS_Glossary, can be found on eRooms. Table 5 contains those terms most applicable for this OAD.

Table 5. Glossary

Term	Description
Algorithm	A formula or set of steps for solving a particular problem. Algorithms can be expressed in any language, from natural languages like English to mathematical expressions to programming languages like FORTRAN. On NPOESS, an algorithm consists of: A theoretical description (i.e., science/mathematical basis) A computer implementation description (i.e., method of solution) A computer implementation (i.e., code).
Algorithm Configuration Control Board (ACCB)	Interdisciplinary team of scientific and engineering personnel responsible for the approval and disposition of algorithm acceptance, verification, development and testing transitions. Chaired by the Algorithm Implementation Process Lead, members include representatives from IWPTB, Systems Engineering & Integration IPT, System Test IPT, and IDPS IPT.
Algorithm Verification	Science-grade software delivered by an algorithm provider is verified for compliance with data quality and timeliness requirements by Algorithm Team science personnel. This activity is nominally performed at the IWPTB facility. Delivered code is executed on compatible IWPTB computing platforms. Minor hosting modifications may be made to allow code execution. Optionally, verification may be performed at the Algorithm Provider's facility if warranted due to technical, schedule or cost considerations.
Ancillary Data	Any data which is not produced by the NPOESS System, but which is acquired from external providers and used by the NPOESS system in the production of NPOESS data products.
Auxiliary Data	Auxiliary Data is defined as data, other than data included in the sensor application packets, which is produced internally by the NPOESS system, and used to produce the NPOESS deliverable data products.
EDR Algorithm	Scientific description and corresponding software and test data necessary to produce one or more environmental data records. The scientific computational basis for the production of each data record is described in an ATBD. At a minimum, implemented software is science-grade and includes test data demonstrating data quality compliance.
Environmental Data Record (EDR)	*[IORD Definition]* Data record produced when an algorithm is used to convert Raw Data Records (RDRs) to geophysical parameters (including ancillary parameters, e.g., cloud clear radiation, etc.). *[Supplementary Definition]* An Environmental Data Record (EDR) represents the state of the environment, and the related information needed to access and understand the record. Specifically, it is a set of related data items that describe one or more related estimated environmental parameters over a limited time-space range. The parameters are located by time and Earth coordinates. EDRs may have been resampled if they are created from multiple data sources with different sampling patterns. An EDR is created from one or more NPOESS SDRs or EDRs, plus ancillary environmental data provided by others. EDR metadata contains references to its processing history, spatial and temporal coverage, and quality.
Model Validation	The process of determining the degree to which a model is an accurate representation of the real-world from the perspective of the intended uses of the model. [Ref.: DoDD 5000.59-DoD Modeling and Simulation Management]
Model Verification	The process of determining that a model implementation accurately represents the developer's conceptual description and specifications. [Ref.: DoDD 5000.59-DoD Modeling and Simulation Management]
Operational Code	Verified science-grade software, delivered by an algorithm provider and verified by IWPTB, is developed into operational-grade code by the IDPS IPT.
Operational-	Code that produces data records compliant with the System Specification requirements for

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

Term	Description
Grade Software	data quality and IDPS timeliness and operational infrastructure. The software is modular relative to the IDPS infrastructure and compliant with IDPS application programming interfaces (APIs) as specified for TDR/SDR or EDR code.
Raw Data Record (RDR)	*[IORD Definition]* Full resolution digital sensor data, time referenced and earth located, with absolute radiometric and geometric calibration coefficients appended, but not applied, to the data. Aggregates (sums or weighted averages) of detector samples are considered to be full resolution data if the aggregation is normally performed to meet resolution and other requirements. Sensor data shall be unprocessed with the following exceptions: time delay and integration (TDI), detector array non-uniformity correction (i.e., offset and responsivity equalization), and data compression are allowed. Lossy data compression is allowed only if the total measurement error is dominated by error sources other than the data compression algorithm. All calibration data will be retained and communicated to the ground without lossy compression. *[Supplementary Definition]* A Raw Data Record (RDR) is a logical grouping of raw data output by a sensor, and related information needed to process the record into an SDR or TDR. Specifically, it is a set of unmodified raw data (mission and housekeeping) produced by a sensor suite, one sensor, or a reasonable subset of a sensor (e.g., channel or channel group), over a specified, limited time range. Along with the sensor data, the RDR includes auxiliary data from other portions of NPOESS (space or ground) needed to recreate the sensor measurement, to correct the measurement for known distortions, and to locate the measurement in time and space, through subsequent processing. Metadata is associated with the sensor and auxiliary data to permit its effective use.
Retrieval Algorithm	A science-based algorithm used to ‘retrieve’ a set of environmental/geophysical parameters (EDR) from calibrated and geolocated sensor data (SDR). Synonym for EDR processing.
Science Algorithm	The theoretical description and a corresponding software implementation needed to produce an NPP/NPOESS data product (TDR, SDR or EDR). The former is described in an ATBD. The latter is typically developed for a research setting and characterized as “science-grade”.
Science Algorithm Provider	Organization responsible for development and/or delivery of TDR/SDR or EDR algorithms associated with a given sensor.
Science-Grade Software	Code that produces data records in accordance with the science algorithm data quality requirements. This code, typically, has no software requirements for implementation language, targeted operating system, modularity, input and output data format or any other design discipline or assumed infrastructure.
SDR/TDR Algorithm	Scientific description and corresponding software and test data necessary to produce a Temperature Data Record and/or Sensor Data Record given a sensor’s Raw Data Record. The scientific computational basis for the production of each data record is described in an Algorithm Theoretical Basis Document (ATBD). At a minimum, implemented software is science-grade and includes test data demonstrating data quality compliance.
Sensor Data Record (SDR)	*[IORD Definition]* Data record produced when an algorithm is used to convert Raw Data Records (RDRs) to calibrated brightness temperatures with associated ephemeris data. The existence of the SDRs provides reversible data tracking back from the EDRs to the Raw data. *[Supplementary Definition]* A Sensor Data Record (SDR) is the recreated input to a sensor, and the related information needed to access and understand the record. Specifically, it is a set of incident flux estimates made by a sensor, over a limited time interval, with annotations that permit its effective use. The environmental flux estimates at the sensor aperture are corrected for sensor effects. The estimates are reported in physically meaningful units, usually in terms of an angular or spatial and temporal distribution at the sensor location, as a function of spectrum, polarization, or delay, and always at full resolution. When meaningful, the flux is also associated with the point on the Earth geoid from which it apparently originated. Also, when meaningful, the sensor flux is converted to an equivalent top-of-atmosphere (TOA) brightness. The associated metadata includes a record of the processing and sources from which the SDR was created, and other information needed to understand the data.
Temperature Data	*[IORD Definition]*

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

Term	Description
Record (TDR)	Temperature Data Records (TDRs) are geolocated, antenna temperatures with all relevant calibration data counts and ephemeris data to revert from T-sub-a into counts. *[Supplementary Definition]* A Temperature Data Record (TDR) is the brightness temperature value measured by a microwave sensor, and the related information needed to access and understand the record. Specifically, it is a set of the corrected radiometric measurements made by an imaging microwave sensor, over a limited time range, with annotation that permits its effective use. A TDR is a partially-processed variant of an SDR. Instead of reporting the estimated microwave flux from a specified direction, it reports the observed antenna brightness temperature in that direction.

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

3.2 Acronyms

The current acronym list for the NPOESS program, D35838_E_NPOESS_Acronyms, can be found on eRooms. Table 6 contains those terms most applicable for this OAD.

Table 6. Acronyms

Acronym	Description
AM&S	Algorithms, Models & Simulations
API	Application Programming Interfaces
ARP	Application Related Product
CDFCB-X	Common Data Format Control Book - External
CMN GEO	Common Geolocation
CSN	Common Short Name
DMS	Data Management Subsystem
DPIS ICD	Data Processor Inter-subsystem Interface Control Document
DQTT	Data Quality Test Table
E&A	Ephemeris and Attitude
ECEF	Earth-Centered Earth-Fixed
ECF	Earth-Centered Fixed
ECR	Earth-Centered Rotating
ECSF	Earth-Centered Space-Fixed
EES	Ellipsoid to Earth's Surface
EGM	Earth Geod Model
EGS	Ellipsoid Geoid Separation
EPHATT	Ephemeris ad Attitude Structure
IET	IDPS Epoch Time is the time in number of microseconds since 1-1-1958 00:00:00
IIS	Intelligence and Information Systems
INF	Infrastructure
ING	Ingest
IP	Intermediate Product
LOS	Line of Sight
LUT	Look-Up Table
MDFCB	Mission Data Format Control Book
MSL	Mean Sea Level
NOVAS-C	Naval Observatory Vector Astronometry System – C version
PRO	Processing
QF	Quality Flag
SAA	South Atlantic Anomaly
SDR	Sensor Data Records
SI	Software Item or International System of Units
STL	Standard Template Library
TBD	To Be Determined
TBR	To Be Resolved
TLE	Two Line Element files following the format specified by NORAD
TOA	Top of the Atmosphere
UT1	Universal Time One
UTC	Universal Time Coordinated

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO

4.0 OPEN ISSUES

Table 7. List of OAD TBD/TBR

No.	DESCRIPTION	Resolution Date
None		

D41869, A. PDMO Released: 2010-02-02 (VERIFY REVISION STATUS)
PDMO